



SSD (Solid-State Drive)를 이루는 기술과 미래

SSD는 고성능, 고신뢰성 등의 여러가지 장점으로 모바일, PC, 데이터센터 등의 다양한 응용 분야에서 빠르게 확산되고 있다. 이는 저장 매체인 플래시메모리의 고용량화 외에 이를 운용하는 SoC (System-on-chip) 그리고 소프트웨어 기술이 통합 최적화에 따른 결과이다. 본 논문에서는 SSD를 발전시켜온 주요 하드웨어 및 소프트웨어 기술요소들을 소개하고, 향후의 진화 방향에 대해서 살펴보기로 한다. 특별히 빅데이터와 클라우드의 등장에 따라 SSD의 활용도는 크게 증가할 것이며, 그 기술은 더욱 더 발전할 것으로 예상된다.

I. 서론

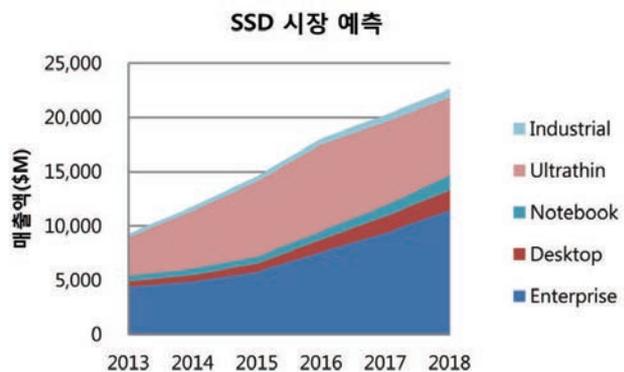
SSD가 고성능, 고신뢰성, 저전력, 경박단소, 경량화의 장점을 가지고, 모바일 기기에서 노트북, 데스크탑 그리고 엔터프라이즈 서버 시스템에 이르기까지 다양한 응용을 위한 저장장치로 사용되고 있다 (그림 1).



정재현
삼성전자



박찬익
삼성전자



〈그림 1〉 SSD 시장 응용별 비중^[1]

〈표 1〉 HDD와 SSD의 특성 비교

	HDD	SSD
부팅 시간(초)	40	22
평균 I/O 처리성능 (IOPS)	410	39000
소모 전력(W)	8.3	3.7
단위전력당 성능(IOPS/W)	49	10405
Mean Time Between Failure(백만시간)	1.5	2

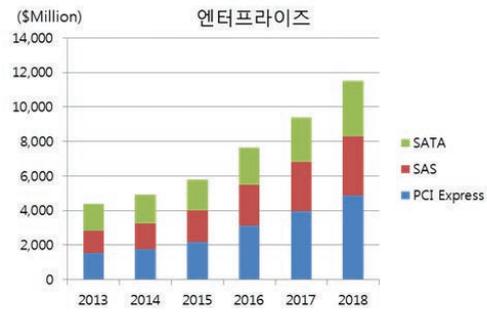
(출처: http://www.storagereview.com/ssd_vs_hdd)

〈표 1〉은 다양한 응용에서 SSD가 HDD대비 갖는 장점을 보여준다.

SSD는 PC/노트북에서는 빠른 부팅과 응용 프로그램의 고성능 수행을 가능하게 하며 주저장장치로서 채용율이 증가하고 있으며, 특히 유희시간 저전력 기술들을 통한 초절전 모드 지원 등은 노트북 및 울트라씬 기기의 필수 구성요소로 작용하고 있다. 엔터프라이즈 시장에서는 서버와 스토리지 시스템에서 접근 빈도가 높은 데이터(hot data)의 저장을 위한 Read 혹은 Write 캐쉬로 활용되어 서버의 데이터 처리 능력을 향상시켰다. 한편 최근 클라우드 및 데이터센터가 급성장하면서 대규모의 서버시스템들을 위한 부팅 드라이브 및 가상화 환경에서 여러 가상 머신을 위한 공유 드라이브로 사용되고 있으며, 고용량, 고속화 추세를 견인하고 있다. HDD 대비 비교적 높은 초기 비용에도 불구하고 엔터프라이즈 시장에서 SSD가 각광받는 이유는 낮은 소

모 전력과 상대적으로 작은 불량률에 따른 교체 비용 감소 등 유지 보수 비용을 고려한 TCO (total cost of ownership)의 효율 때문이다^[2].

SSD는 SATA, SAS, PCI Express (PCIe) 등의 물리적 호스트 인터페이스에 따라 그 성능과 신뢰성, 가격 그리고 응용처가 구분된다. SATA는 비교적 간단한 구조를 채택하고 있어서 저가형 드라이브 인터페이스로 적합하여 클라이언트 환경에서 주로 채택되어 사용되며, SAS는 고신뢰성을 지원하기 위한 기능들이 구현되어 있어서 서버나 스토리지 시스템과 같은 엔터프라이즈 환경에 적합하다. 최근 시스템 버스 규격인 PCIe를



〈그림 2〉 SSD 인터페이스별 시장 규모^[1]

물리적 인터페이스로 가지는 SSD가 빠른 응답시간과 성능 확장성의 장점을 가지고 데이터베이스 응용 등 고사양 엔터프라이즈 시스템에서 채용되기 시작되었으며 점차 그 비중을 빠르게 확대해 나갈 것으로 예상된다 〈그림 2〉.

클라이언트 환경에서도 PCIe기반의 SSD가 고사양 노트북 등에서 사용되기 시작하며 시장에 진입하고 있으나, 중저가의 노트북과 울트라북 등의 시스템에서는 SATA가 3Gb/s에서 6Gb/s로 전환되어 큰 비중을 차지할 것으로 예상된다.

이렇게 빠르게 SSD가 HDD를 대체 혹은 새로운 응용 시장에 진입, 확대되고 있는 기술적 배경은 무엇일까? 그것은 바로 고집적도 플래시메모리, 고성능 컴퓨팅을 위한 SoC 기술 그리고 이를 운용하는 소프트웨어 기술의 융합에서 기인한다. 본 논문에서는 이러한 기술적 발전의 과정을 SSD 구성 요소별로 살펴보고, 향후 어떻게 SSD가 발전하여 나갈 지를 기존 연구와 제품 등을 통해 살펴보기로 한다.

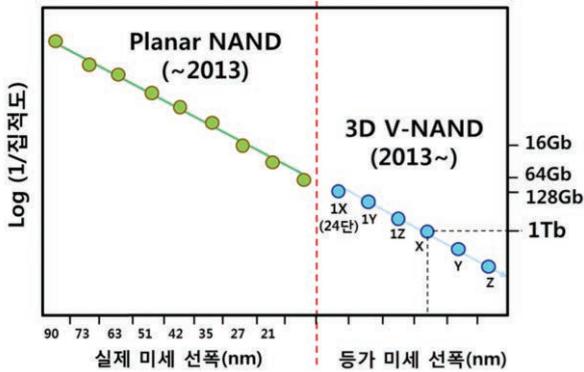


II. 본론

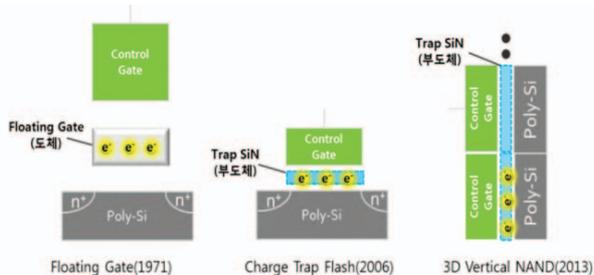
2.1. 플래시메모리

SSD가 대중화될 수 있었던 가장 큰 기술적 혁신으로 저장요소인 플래시메모리의 고집적화를 들 수 있다. 거의 매년 2배에 가까운 집적도 증가를 보여주며 90년 이후 technology shrink로 인한 고집적화를 지속해 왔다. 여기에 멀티레벨 셀기술의 발전으로 동일한 셀에 2비트 그리고 3비트까지 저장하는 기술이 상용화됨으로써 고용량 SSD 시장을 가능하게 하였다. 셀 집적도가 고도화되면서 셀간 간섭으로 인한 에러 비트율이 증가하는 등 기술의 한계를 맞는 듯 하였으나, 스케일링과 설비 한계 기술을 극복하는 3D 낸드 기술이 나오므로써 다시 한번 고용량화의 추세를 이어나갈 수 있게 되었다. 그 결과 칩당 512기가비트와 1테라비트 이상의 제품 개발이 가능하게 할 예정이다<그림 3>.

기존 Floating Gate기반의 Planar NAND cell은



<그림 3> NAND 고집적화의 지속^[3]



<그림 4> NAND Cell의 발전^[3]

Floating Gate에 전하를 저장하는 방식을 사용하였다<그림 4>. 그러나 최근 10나노급 공정 도입시 셀간 간격이 좁아질수록 전자가 누설되는 간섭 현상이 심화되는 문제가 발생하게 되어 집적도의 증가가 한계에 부딪히게 되었다.

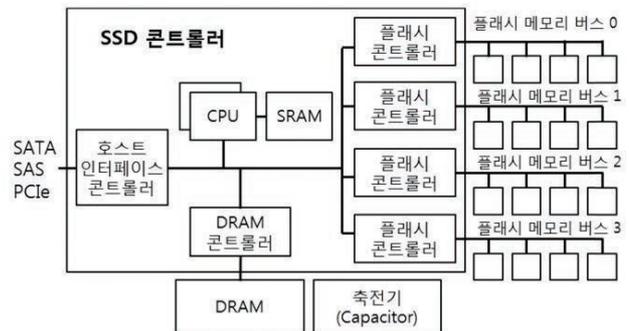
이를 극복하기 위한 기술로 전하를 부도체에 저장하는 저장하는 방식인 Charge Trap Flash(CTF) 기술이 개발되었고, 이를 기반으로 3차원 형태로 적층하여 나온 기술이 바로 3D 수직 낸드 플래시메모리이다. 기존 설비 공정 등을 거의 그대로 사용하면서 2차원의 공간적 제약을 3차원으로 해결함으로써 셀간의 간섭을 줄였기 때문에 용량이 증가하면서도 신뢰성 및 성능을 향상할 수 있게 되었다. 이는 SSD 응용을 보다 고용량, 고신뢰성을 요구하는 분야로 확대하는 효과를 가져올 것으로 기대한다.

SSD는 SSD 컨트롤러, 다수의 플래시메모리 그리고 DRAM으로 구성된다.

2.2. SSD 하드웨어

SSD가 고속의 성능을 보이면서 저전력 장점을 갖게 한 것은 SoC (System-on-Chip) 기술의 힘이 크다. SSD는 <그림 5>에서 보는 바와 같이 SSD 컨트롤러, 다수의 플래시메모리 그리고 DRAM으로 구성된다. 응용시스템 요구사항에 따라 정전시 데이터 손실을 막기 위해 PLP(power-loss protection)를 위한 축전기를 추가적으로 사용하기도 한다.

우선 호스트로부터 들어오는 읽기, 쓰기 명령을 처리



<그림 5> SSD의 구성요소와 SSD 컨트롤러의 구조

하기 위한 호스트 인터페이스 컨트롤러가 있는데, 호스트로부터 명령어를 받아서 데이터가 읽혀지거나 쓰여질 논리 주소를 해석하고, 데이터의 양을 결정하도록 한다. 또한 호스트 혹은 플래시 메모리로부터 이동된 데이터를 임시로 저장하기 위한 버퍼메모리 관리를 수행하기도 한다. 특히 호스트 인터페이스의 성능은 물리적 대역폭의 증가와 함께 동시에 처리할 수 있는 명령어 수의 증가를 통해 향상되고 있다. 동시에 명령어를 처리할 수 있는 명령어 대기열을 Command Queuing이라고 하고, 이는 SSD의 내부 플래시메모리의 병렬성을 최대 활용하는데 기여 한다.

호스트인터페이스의 고속화 및 병렬성 증가에 대응하여 SSD는 플래시메모리의 병렬성을 활용하는 아키텍처를 구현하고 있다. 플

래시 메모리들을 독립적인 플래시 메모리 버스(채널)들에 연결하여 이를 멀티 채널구조라 부르고, 이 독립적인 채널을 다시 여러 플래시메모리가 공유하는 멀티웨이 인터리빙 구조가 복수개의 플래시메모리를 구성하는 대표적인 아키텍처가 된다. 이는 호스트인터페이스를 통해 들어오는 명령어가 최대한 병렬적으로 여러 플래시메모리에서 데이터를 읽고 쓸 수 있도록 함으로써 고속의 데이터 I/O를 가능하게 한다. 기계적인 회전 디스크에서 헤드를 통해 순차적으로 데이터를 액세스하는 HDD하고는 확연히 다른 구조가 아닐 수 없다.

하나의 독립 채널을 제어하기 위한 플래시 컨트롤러는 ECC(error correction code)와 플래시 메모리에서 데이터를 DRAM으로 전송하는 DMA 기능을 포함하고 있으며 또한 이를 운용하는 소프트웨어 알고리즘이 실행되기도 한다. DRAM은 플래시 메모리에서 데이터를 호스트로 보내기 위한 임시 버퍼 메모리로 사용되는 것 외에 호스트시스템이 바라보는 논리적 주소를 플래시메모리의 물리적 주소로 변환하기 위한 매핑 테이블을 저장하는데 사용되기도 한다. 매핑 방법에 따라 그 메모리 요구량은 플래시메모리 GB당 MB 수준의 DRAM이 필요로 하며 SSD의 고용량화됨에 따라 DRAM의 성능 및 용량 최적화가 기술적 도전이 되고 있다.

SSD 하드웨어는 고성능, 저전력, 확장 가능성을 고려한 시스템 기술들이 적용되어 개발되어 오고 있다.

엔터프라이즈 환경의 스토리지나 서버 시스템처럼 데이터 보존의 요구사항이 중요한 응용에서는 DRAM과 같이 휘발성메모리에 저장된 사용자 데이터와 매핑 데이터를 갑작스런 정전 시에 플래시 메모리에 저장하기 위해 작은 용량의 축전기를 사용하기도 한다.

ECC는 플래시메모리에서 데이터 읽기 동작 시 발생할 수 있는 비트 에러를 정정해 주기 위한 기술로서 Hamming code, RS(Reed-Solomon) 및 BCH(Bose-Chaudhuri-Hocquenghem)등 고전적 에러 정정 코드에서 LDPC(Low Density Parity Check)와 같은 보다

높은 에러 정정율을 가지는 코드로 발전하고 있다. ECC의 높은 계산량과 패리티 용량은 SSD의 에너지 소모 및 플래시메모리 비용 경쟁력에 영향을 주기 때문에

보다 지능화된 ECC 알고리즘과 하드웨어 구조가 지속적으로 연구 발전되고 있다.

하드웨어 구성의 마지막 요소로 멀티 CPU 아키텍처를 언급하기로 한다. 명령어 대기열에 들어오는 다수의 호스트 명령어를 해석하여 복수의 플래시 컨트롤러를 통해 수십 개의 플래시메모리에서 데이터를 읽고 쓰기 위해서는 높은 수준의 컴퓨팅 파워가 필요하게 되며, 이를 위해 고속의 멀티 CPU 구조가 채용되게 된다. 멀티 CPU들간에는 호스트의 명령어 처리, 버퍼 메모리 및 매핑 테이블 관리 그리고 플래시 메모리로의 데이터 전송 제어 등을 담당하도록 태스크가 고정 혹은 동적으로 분할되어지기도 한다. SRAM은 멀티 CPU에서의 펌웨어 수행을 위한 코드 및 데이터 저장용으로 사용된다.

결론적으로 SSD 하드웨어는 다수의 플래시메모리,

〈표 2〉 SSD 펌웨어(firmware)의 주요 기능

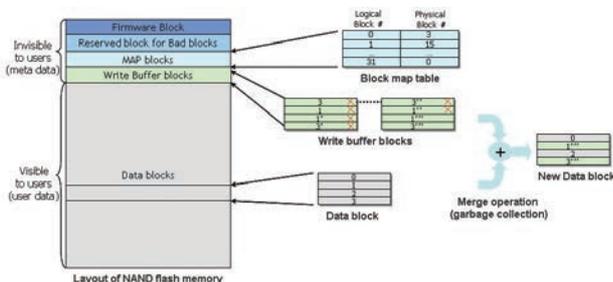
기술	주요 기능
주소 매핑	호스트의 논리적 주소를 플래시 주소로 변환
가비지컬렉션	무효한 데이터를 삭제하고 Free 블록 생성
정전 복구	갑작스런 정전시 이전 데이터 복구
배드블록관리	초기 혹은 진행성 배드 블록의 처리
웨어레벨링	블록별 삭제 수의 균형을 유지하여 수명제고
실시간성 지원	호스트 명령어를 주어진 시간 안에 처리

이를 운용하기 위한 SoC기반 컨트롤러, DRAM과 축전기 등이 조합된 하나의 복합 시스템으로서 고성능, 저전력, 확장 가능성을 고려한 시스템 기술들이 적용되고 개발되어 오고 있다.

2.3. SSD 펌웨어

〈표 2〉에서 보는 바와 같이 SSD는 다양한 펌웨어의 기능을 필요로 한다. 주요 기능중 하나로 호스트의 논리적 주소를 SSD내 플래시메모리의 물리적 주소로 변환하는 매핑 기술이 있다. 이는 플래시메모리가 HDD와는 달리 데이터에 대한 덮어쓰기(overwrite)가 허용되지 않는 물리적 특성 때문에 발생하는 제한을 극복하는 기술로서 성능과 수명 그리고 메모리 요구량에 가장 큰 영향을 주는 기술 요소이다. 이에 대한 다양한 연구와 성능 분석은 기존 연구에서 찾아볼 수 있다^[4-5].

SSD의 동작을 통해 주요 펌웨어 기술들을 살펴보기로 하자. 우선 〈그림 6〉의 왼쪽에서 보는 바와 같이 플래시메모리는 사용자 데이터를 저장하기 위한 데이터 블록 외에 수행될 코드를 저장하는 펌웨어 블록(firmware block), 배드 블록이 발생했을 때 이를 대체하기 위한 예비 블록(reserved block), 매핑 테이블을 저장해 놓은 맵 블록(MAP block), 그리고 쓰기 버퍼 블록(write buffer block)으로 구성되어 있다. 맵 블록의 경우 논리 주소와 물리 주소의 변환을 위한 주소값을 저장하고 있으며, 시스템 부팅시 SSD내 메모리에 적재되어 플래시메모리로의 쓰



〈그림 6〉 플래시메모리의 구성과 쓰기 동작

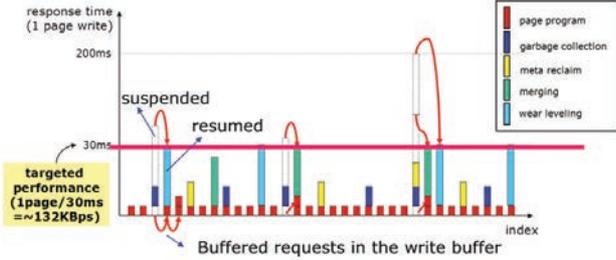
기 동작이 발생할 때마다 갱신이 일어나게 된다. 쓰기 버퍼 블록은 미리 확보해 놓은 삭제 블록(erasured block)으로서 호스트 데이터를 받아 저장하는 역할을 한다. 만일 쓰기 버퍼 블록이 모두 소진되었을 경우 새로운 쓰기 버퍼 블록 확보를 위해 가비지 컬렉션(garbage collection)이 일어나게 되는데, 이는 데이터 블록과 쓰기 버퍼 블록에서 유효한 데이터만 가져다가 새로운 블록에 모아 쓰는 합병 연산(merge operation)에 해당한다. 그 결과 새로운 삭제 블록이 생성되게 된다. 가비지 컬렉션은 SSD 동작 중 가장 시간이 오래 걸리는 연산이며, 매핑 기술과 더불어 SSD 성능과 수명에 가장 큰 영향을 주는 동작이라 이를 줄이기 위한 다양한 기술들이 연구되어 왔다. 특히 데이터를 접근 속성에 따라 관리하는 hot/cold 구분 기술이 가비지 컬렉션의 효율, 즉 성능을 결정하는 요소로서 지속적으로 개선되고 있다.

SSD 펌웨어의 주요 기능은 주소 매핑, 가비지컬렉션, 정전 복구, 배드블록관리, 웨어레벨링 및 실시간성 지원 등이다.

그외 SSD의 신뢰성을 위한 기술들을 살펴보면 갑작스런 정전시 데이터 복구 기술(power-off recovery)를 들 수 있다. 언제든 정전이 될 수 있다는 것을 고려하여 중요 데이터(예를 들어 매핑

데이터)는 변경점이 있을 때마다 플래시메모리에 최신의 것으로 업데이트를 유지하도록 한다. 또한 비정상 종료후 부팅(booting)이 이루어질 때 이전에 수행했던 작업을 완료하거나 복구하는 기술이 필요한데 이를 위해 파일시스템에서 메타데이터 복구를 위해 사용해 오고 있는 저널링(Journaling) 기법과 유사한 방법이 적용되기도 한다. 서버형 SSD에서는 축전기 등을 이용하여 정전시 DRAM에 있는 사용자 데이터를 플래시메모리에 보존하는 기능을 제공하기도 한다.

플래시메모리의 특성 중 하나는 블록당 제한된 쓰기/삭제 수를 가지고 있다는 것이다. 따라서, 자칫 하나의 블록에 쓰기가 집중되는 경우 블록 수명이 소진(wear-out)되는 문제가 발생할 수 있다. 이를 방지하기 위해 플래시메모리 전체에 걸쳐 평준화된 삭제 수를 갖도록 관리하는 웨어레벨링 기술이 적용된다. 블록당 삭제 수



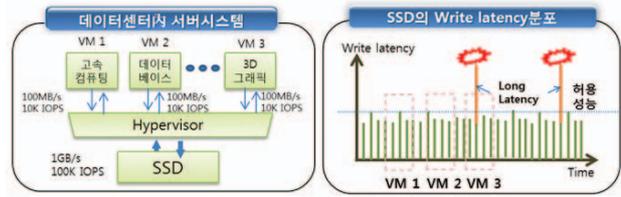
〈그림 7〉 응답시간 관리 기술

를 저장하여 이를 변경 때마다 최소에서 최대값까지 정렬하여 가장 적게 쓰여진 블록과 많이 쓰여진 블록을 교체하여 사용하도록 하는 정적 방법과 사용자 워크로드의 액세스 패턴(예를 들어 순차적 쓰기)에 의존하여 블록들이 균등하게 쓰여지고 삭제되도록 하는 동적 방법이 응용 특성 및 메모리 요구량에 따라 적용되고 있다.

최근 사용자 스토리지 디바이스의 성능 체감에서 중요하게 여겨지는 요구사항으로 실시간성이 중요해 지고 있다. 위에서 설명한 가비지 컬렉션과 웨어레벨링 등이 간헐적이지만 발생할 경우, 비교적 긴

시간을 요구하는 연산이기 때문에 사용자 응답시간이 길어지는 문제가 발생할 수 있기 때문이다. 예를 들어 〈그림 7〉에서와 같이 목표 응답시간(targeted performance)이 30ms 이내라고 할 때, 일반적인 데이터 쓰기는 그 응답시간이 충분히 이 요구조건을 만족시키지만, 비교적 긴 수행시간을 요구한 연산이 발생하는 경우 30ms를 넘어 200ms까지 길어질 수 있다. 이를 해결하기 위해 길어지는 연산이 종료될 때까지 기다리지 않고, 호스트의 새로운 쓰기 혹은 읽기 요구를 받아들여 호스트의 처리가 지연되지 않도록 하는 기술이 필요하다. 이를 구현하는 방법으로 연산의 중단/재개, 단계적 가비지 컬렉션, 실시간 스케줄링 기법 등이 연구되어 왔다.

이러한 일관된 빠른 응답시간성, 즉 QoS의 중요성은 SSD가 데이터센터내 클라우드 환경에서 사용되면서 더욱 증대되고 있다. 〈그림 8〉과 같이 서버내 SSD는 여



〈그림 8〉 클라우드 환경에서의 QoS

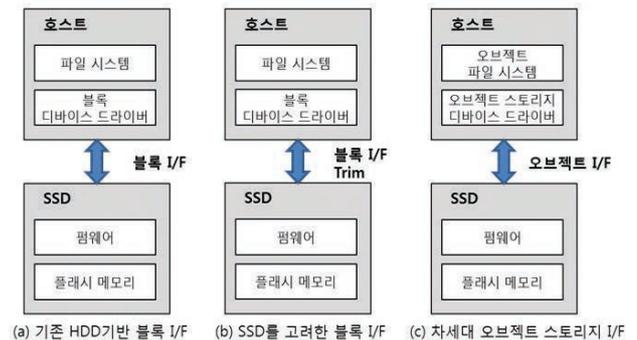
러 가상 머신에 의해 공유가 되는 스토리지이며, 이를 접근하는 동안 각 가상 머신은 서비스 계약에서 약속된 성능을 보장받아야 한다(SLA: service level agreement). 그러나, 위에서 설명한 긴 연산 시간(long latency)이 갑자기 발생하여 해당 가상 머신이 상대적인 성능 저하를 경험하게 된다면 공정한 서비스 관점에서 손해를 보게 되며, 이는 클라우드 서비스 업체의 비즈니스에도 큰 영향을 줄 수 있다.

SSD가 데이터센터내 클라우드 환경에서 사용 되면서 일관된 빠른 응답시간성 (QoS: Quality of Service)의 중요성이 증대되고 있다.

2.4. 호스트 인터페이스

마지막으로 SSD의 기술 진화와 함께 주목할 것은 논리적 호스트 인터페이스의 변화이다. 블록 인터페이스 기반 HDD의 소

프트웨어 스택을 그대로 활용하여 호스트내 응용프로그램과 파일시스템의 수정 없이 SSD가 HDD인 것처럼 사용된 것은 SSD의 성공적인 시장 진입을 가능하게 하였다(그림 9(a)). 그러나, HDD에 최적화된 에코시스템에서 SSD의 성능 체감은 크게 제한이 되었다. 특히, 플래시메모리가 가진 특성, 즉 블록을 삭제해야 다시

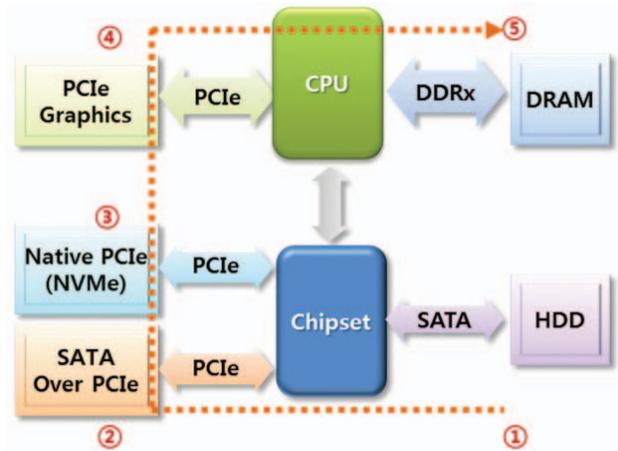


〈그림 9〉 논리적 호스트인터페이스의 진화

쓰기를 할 수 있는 것 때문에 모든 삭제 블록이 소진된 이후에는 성능이 급격히 떨어지는 단점이 노출되었다. 이를 극복하기 위해 SSD의 이러한 삭제 후 쓰기 가능 특성을 고려한 인터페이스 변화가 있었는데, 호스트 운영체제(파일시스템)에서 무효화된 데이터의 영역을 스토리지 디바이스에게 알려주는 Trim 명령어의 지원이 그것이다^[6]. 이 새로운 인터페이스 커맨드를 통해 파일 시스템은 할당한 영역 중 어느 부분이 사용되지 않는다는 것을 명시적으로 SSD에 전달함으로써 SSD가 해당 영역(블록)을 삭제하여 새로운 쓰기 블록으로 사용할 수 있도록 하였다(그림 9(b)). 이를 통해 랜덤(random) 쓰기 성능이 급격히 향상될 수 있었다. 더 나아가 호스트 파일시스템 기능 중 하나인 주소변환 기능을 SSD내부로 옮기고, 데이터 속성 정보를 SSD에 전달해 줄 수 있는 오브젝트 스트리지(OSD) 인터페이스 개념이 나왔으며^[7], 이는 데이터의 저장 위치를 SSD내부에서 결정하도록 하여 최대한 같은 속성(예를 들어 hot또는 cold)을 가진 데이터를 같은 블록 안에 포함되도록 하여 가비지 컬렉션 효율을 높이며, 이로 인해 수명을 향상시킬 수 있도록 하였다. 게다가 데이터의 속성을 SSD내에 전달하여 이를 고려한 매핑, 웨어레벨링, 실시간 처리 하는데 SSD의 컴퓨팅 능력과 내부 리소스 등을 활용하도록 하는 지능형 스토리지 디바이스의 가능성을 제안하게 하였다(그림 9(c)).

2.5. 향후 기술 진화 방향

지금까지 SSD 대중화를 가능하게 했던 SSD관련 하드웨어, 소프트웨어 그리고 인터페이스의 변화 과정 등에 대해 살펴보았다. 향후 SSD의 진화 방향을 살펴보기 위해서는 현재 CPU기반 시스템에서의 SSD의 발전 과정과 최근 급진적으로 변화하고 있는 엔터프라이즈 기술 환경 변화를 고려하는 것이 필요하다. <그림 10>에서 보는 바와 같이 SSD는 호스트 시스템내에서 SATA HDD의 대체로 시작하여 그 특유의 고성능을 활용하기 위해 이보다 고속의 PCIe 기반 인터페이스로 그 위치를



<그림 10> 시스템내 SSD의 발전 방향

옮기고 있다. 이는 더 나아가 가장 빠른 인터페이스에 해당하는 메모리 인터페이스로의 진화가 가능하다는 것으로 예상할 수 있으며 실제로 이를 지향하는 SSD 제품들이 최근 시장에 선을 보이기 시작하였다. 빅데이터 및 클라우드 환경에 대응하기 위한 데이터센터의 급성장으로 인해 분산 자원의 효율적 관리와 시스템내 병렬성 활용의 극대화를 위해 가상화 기술이 더욱 중요해지고 있다. 멀티 코어와 가상 머신 기술이 도입된 지 오래되었지만 스토리지 등의 병목으로 인해 큰 성능 이익을 얻기는 어려웠는데, SSD의 도래에 따라 이러한 멀티 코어 병렬성 그리고 가상화 기술 활용이 시스템 효율 및 성능 최적화에서 크게 기여하고 있다. 이를 지원하기 위해 최근 발전되고 있는 고속의 차세대 스토리지 인터페이스가 NVM Express(NVMe) 이다.

2.5.1. NVMe: 차세대 호스트 인터페이스

고속, 고용량 SSD 확산 그리고 이러한 멀티코어 기반 가상화 환경을 대응하기 위해 PCIe기반의 NVMe 인터페이스가 표준화되어 제품으로 출시되기 시작했다^[8]. NVMe가 기존 SATA 인터페이스대비 가지는 장점은 동시 처리 가능한 명령어 큐 및 큐에 포함되는 명령어 수의 증가와 인터럽트의 처리 효율화 그리고 빠른 응답 시간이다. 병렬성을 고려한 인터럽트 처리는 기존

멀티 코어 병렬성 및 가상화 기술 활용이 시스템 효율 및 성능 최적화에 크게 기여하고 있다.

SATA에서는 스토리지 디바이스에서 발생하는 인터럽트가 한 코어에서 하나의 큐 구조를 가지고 처리되는 것에 비해 NVMe의 경우 여러 코어에서 멀티 큐 구조를 가지고 동시에 여러 개의 명령어 처리가 가능하도록 지원한다(그림 11).

NVMe의 또 다른 특징은 서버 가상화를 지원하기 위한 SR-IOV (single root IO virtualization) 그리고 복수개의 Namespace 지원이다. 먼저 SR-IOV는 서버환경에서 가상 머신들이 스토리지 디바이스를 전용 스토리지인 것처럼 사용할 수 있게 접근시 주소를 변환해주는 역할을 한다. 현재는 하이퍼바이저(hypervisor)를 통해 소프트웨어적으로 이루어지는 일이지만 이를 SSD 내에서 하드웨어로 지원함으로써 성능 향상의 결과를 가져올 것으로 기대하고 있다.

Namespace는 가상 머신별로 독립된 주소공간을 할당하도록 하여 이에 대한 배타적 접근을 가능하도록 한다. 이를 통해 가상 머신별 액세스 패턴에 최적화된 SSD의 성능 및 신뢰성 최적화가 가능하도록 할 수 있다. 예를 들어 읽기 중심적인 Namespace의 할당이 있는 경우 메모리 효율적인 매핑 기술을 사용할 수 있으며, 쓰기 중심적인 응용의 경우 쓰기 성능에 최적화된 매핑 기술을 사용할 수 있도록 할 수 있을 것이다. 결과적으로 SR-IOV와 Namespace 기술을 통

해 스토리지 가상화를 위한 자원 효율성을 극대화 할 수 있을 것으로 보인다.

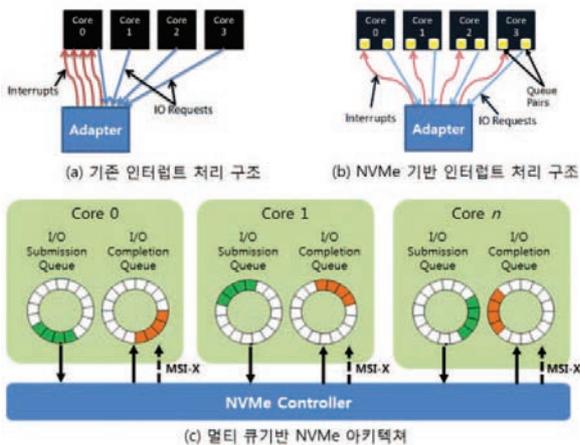
2.5.2. DRAM 인터페이스를 가진 SSD

최근 DRAM 모듈 형식의 DIMM 기술을 SSD에 적용하는 사례가 선을 보이고 있다. 이는 고속의 DRAM 인터페이스를 통해 보다 개선된 응답 시간 효과 그리고 시스템내 DRAM 모듈용 슬롯을 활용할 수 있다는 공간 활용성 때문에 일부 응용에서 채용이 되기 시작하고 있다. 이는 다시 크게 세가지 타입으로 구분이 가능한데 첫번째는 NVDIMM(Non-volatile DIMM)으로서 DRAM, 플래시메모리 그리고 큰용량의 축전기로 구성되어 DRAM의 성능을 보여줌으로써 반응시간이 중요한 메모리 기반 데이터베이스 등에서 활용도가 높다. 그러나, 여기서 플래시메모리는 DRAM의 데이터 백업용도 일뿐 SSD로서의 기능을 한다고 보긴 어렵다. 두번째는 DRAM 인터페이스를 가진 SSD로서 물리적으로는 DRAM 인터페이스를 사용하지만 논리적으로는 블록 기반 인터페이스이므로 기존 SSD와 동일한 메커니즘을 갖는다. 마지막으로 DRAM과 플래시메모리를 하이브리드하게 사용하여 응용 워크로드의 접근 지역성을 고려하여 hot데이터는 DRAM에 캐싱이 되고, cold데이터는 플래시메모리에 저장되도록 하여, 마치 성능은 DRAM에 가까우면서 용량은 플래시메모리에 준하는 대용량 메모리로 사용되도록 하는 연구가 활발하다.

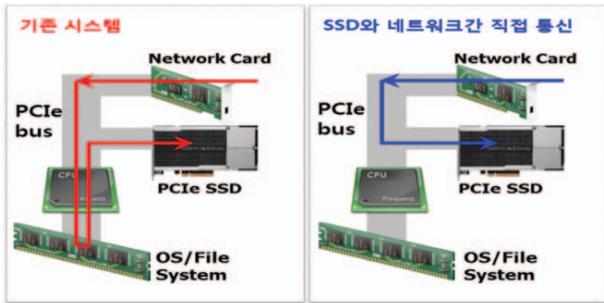
DRAM과 플래시메모리의 융합과 더불어 SSD와 네트워크, 혹은 GPU 기능의 통합 또한 새로운 기술 방향을 제시하고 있다.

2.5.3. PCIe기반 직접(peer-to-peer) 통신

DRAM과 플래시메모리의 융합과 더불어 SSD와 네트워크, 혹은 GPU 기능의 통합 또한 새로운 기술 방향을 제시하고 있다. 최근 논문 결과에 따르면 네트워크와 SSD가 하나로 병합이 되었을 경우 그 응답시간은 10분의 1로 향상된다는 연구결과를 내놓은 바 있다^[10]. 향후 데이터 이동과 계산량이 증가하면서 이러한 반응 시간 효율적인 디바이스간 연동 기술이 주목을 받을 것



〈그림 11〉 NVMe의 멀티 큐 아키텍처^[9]



〈그림 12〉 PCIe 디바이스간 직접 통신 구조

이라 예상된다^[11].

SSD가 PCIe 버스에 연결됨으로써 기존 PCIe에 연결되어 있던 네트워크카드, 그래픽카드(GPU)와의 직접(peer-to-peer) 커뮤니케이션을 통한 응답시간, 대역폭 최적화가 시도되고 있다.

예를 들어 〈그림 12〉의 왼쪽에서 보는 바와 같이 기존 데이터 경로는 네트워크 카드를 통해 들어온 데이터가 시스템 버스를 거쳐 CPU의 개입을 통해 시스템 메모리로 이동하게 되고, 파일시스템을 통해 최종적으로 PCIe SSD에 저장이 이루어지게 된다. 반면 〈그림 12〉의 오른쪽에서는 데이터가 네트워크 카드를 통해 들어와서 바로 PCIe SSD로 저장이 되는 형태가 가능하다. 이 경우 데이터 이동 경로를 줄이는 결과로 응답 시간이 대폭 짧아지고 데이터 이동거리가 줄어든 데 따른 에너지 소모 절감도 가능하게 된다. 데이터 양이 폭증하는 데이터센터에서 성능과 에너지 효율을 높일 수 있는 기술로 주목된다.

2.5.4. 데이터 보안

최근 정보보호 이슈가 사회적으로 부각되면서 공공기관에서의 일정기간 데이터 보존 의무화, 정보 보호를 위한 데이터 폐기 및 접근 관리 효율성이 보다 더 중요해지고 있다. 이를 위한 SSD수준의 보안 기술로 SED (Self-Encryption Drive) 기능이 부각되고 있으며, SED를 위한 TCG (Trusted Computing Group)의 표

준 기술이 SSD에서도 채택되기 시작하였다^[12]. SED 기술이 적용되는 경우, SSD 컨트롤러내 AES (advanced encryption standard) 하드웨어 엔진을 이용하여 플래시메모리로 쓰여지는 데이터를 스토리지 내부에만 존재하는 키로 암호화하며, 인증 절차를 거치지 않으면 읽기와 쓰기가 불가능하다. 설령 분실이나 도난 등을 통해 스토리지 디바이스가 악의적 사용자에게 노출되더라도 인증키를 알지 못하는 경우 접근이 불가능하다. 한편 TCG 표준에 의하면 논리주소 영역별 접근 권한 설정, 부팅 전 인증 절차, Crypto Erase (암호키만 삭제하는 것으로 전체 혹은 일부 파티션의 데이터를 안전하고 빠르게 삭제하는 기능) 등을 위한 호스트와 스토리지간 프로토콜이 제공된다. 마이크로소프트사는 이런 SED와 TCG의 장점을 채택하여 BitLocker에서 SED를 지원하는 기능인 eDrive를 Windows 8 이후 버전부터 포함시키기도 했다^[13].

III. 맺음말

지금까지 SSD가 보편화를 이루게 된 기술적 배경과 향후 발전 방향을 살펴보았다. SSD가 성공적으로 시장에 진입하고 확대된 것은 기존 HDD의 에코시스템을 그대로 활용하였기 때문이다. 그러나, 이는 다시 에코시스템의 제한으로 인해 SSD의 성능을 충분히 사용자에게 전달하지 못하는 한계에 이르렀다. 이를 극복하기 위해 고속 인터페이스인 NVMe이 표준화되고, DRAM 인터페이스를 가진 SSD등이 제품화 되고 있다. 또한 차세대 아키텍처로서 스토리지의 지능화를 위한 인터페이스의 발전 및 시스템내 디바이스간의 협력과 통합을 통해 응답 시간과 대역폭 효율을 높이는 연구도 진행 중이다. 이러한 기술들의 성공적인 안착을 위해서는 무엇보다 응용에서부터 운영체제 그리고 스토리지 디바이스를 아우르는 에코시스템 구축이 필수적이다. 결국 기술과 시장을 차지할 최종 승자는 성공적인 에코빌더일 것임에 틀림없다.

SSD 수준의 보안 기술로 SED (self-encryption drivew) 기능이 부각 되고 있으며, SED를 위한 TCG (trusted computing group)의 표준 기술이 SSD에서도 채택되기 시작하였다.

참고 문헌

- [1] <https://technology.ihs.com/483782/ssd-and-hdd-storage-market-tracker-q1-2014>
- [2] <https://www.snia.org/forums/sssi/programs/TCOcalc>
- [3] <http://www.flashmemorysummit.com/English/Conference/Keynotes.html>
- [4] J. Kim, J. M. Kim, S. Noh, S. L. Min, and Y. Cho, "A space-efficient flash translation layer for compactflash systems," Consumer Electronics, IEEE Transactions on, vol.48, no.2, pp.366-375, May 2002.
- [5] D. Jung, J.-U. Kang, H. Jo, J.-S. Kim, and J. Lee, "Superblock FTL: A superblock-based flash translation layer with a hybrid address translation scheme," ACM Transactions on Embedded Computer Systems, vol. 9, no. 4, pp. 40:1-40:41, March 2010.
- [6] F. Shu and N. Obr, "Data set management commands proposal for ATA8-ACS2," INCITS T13/e07153r6 (Revision 6), December 2007.
- [7] Young-Sik Lee, Sang-Hoon Kim, Jin-Soo Kim, Jaesoo Lee, Chanik Park, Seungryoul Maeng, "OSSD: A case for object-based solid state drives," pp.1-13, IEEE 29th Symposium on Mass Storage Systems and Technologies (MSST), 2013
- [8] http://www.storagereview.com/samsung_announces_industry_s_first_25inch_nvme_ssd
- [9] <http://www.nvmexpress.org/presentations>
- [10] Adrian M. Caulfield and Steven Swanson, "QuickSAN: A Storage Area Network for Fast, Distributed, Solid State Disks", pp. 464-474, Proceeding of the 40th Annual International Symposium on Computer Architecture(ISCA), 2013.
- [11] <https://developer.nvidia.com/gpudirect>
- [12] <http://www.trustedcomputinggroup.org>
- [13] <http://msdn.microsoft.com/en-us/library/windows/hardware/br259095.aspx>



정재현

2001년 University of Southern California, Electrical Engineering (박사)
 1993년 University of Southern California, Electrical Engineering (석사)
 1985년 Korea University, Electronics Engineering (학사)
 2010년 10월~현재 삼성전자 메모리사업부 솔루션 개발실, 연구임원 (전무)
 2005년 7월~2010년 9월 Intel, Server Processor/Platform Architect
 2001년 5월~2005년 7월 IBM, Performance Architect
 1984년 12월~1992년 7월 삼성전자 컴퓨터사업부

<관심분야>
 Computer Architecture, Memory Hierarchy, Storage System, SoC Design, Performance Evaluation, Rapid Prototyping, Software Engineering



박찬익

1995년 서울대학교 컴퓨터공학과 졸업 (학사)
 1997년 서울대학교 컴퓨터공학과 졸업 (석사)
 2002년 서울대학교 전기·컴퓨터공학부 졸업 (박사)
 2002년 3월~현재 삼성전자 메모리사업부 근무

<관심 분야>
 플래시 메모리 기반 스토리지 시스템, 차세대 메모리 아키텍처